

## **AMENDMENT TO THE DRAWINGS**

The attached replacement sheets include changes to Figures 1(a), 1(b), 3(a), and 9(b). The sheets which include Figures 1(a) – 9(b) replace the original sheets including Figures 1(a) – 9(b).

The Figures have been amended to comply with 37 C.F.R. 1.21(d). Applicant respectfully submits that the proposed amendments to the drawings do not add new matter.

**REMARKS**

Applicant respectfully requests reconsideration of this application as amended. Claims 1-23 and 25 are pending in the application. Claims 1-22 and 25 have been amended. Claim 23 has been cancelled. Claims 26 and 27 have been added.

The Examiner objected to the drawings. In response thereto, Applicant herein submits amended drawings in the form of red-inked originals. However, the Examiner specified a number of problems with the drawings. For example, the Examiner indicated that Figure 3(b) has a phrase "Stat Transiti o n" on it left side. Upon review of Figure 3(b), Applicant cannot find the phrase anywhere on Figure 3(b). Furthermore, Applicant cannot find any other missing letters. Even so, Applicant is submitting amended drawings.

The Examiner objected to Claim 22 due to informalities. Applicant has amended Claim 22 to correct its dependency. Applicant respectfully requests the Examiner withdraw the objection.

The Examiner rejected claims 1-23 and 25 under 35 U.S.C. § 102(b) as being anticipated by Mergard (US 6,327,508 B1). Applicant respectfully disagrees.

Claim 1 as amended sets forth the following:

A hardware architecture for the implementation of computational and control flow graphs, the architecture comprising:

a plurality of node elements, wherein each of the plurality of node elements represents a node of a control flow graph;

a plurality of interconnections to connect node elements; wherein each interconnection in the plurality of interconnections represents a distinct transition in the control flow graph;

a plurality of connectivity control logic to independently enable and disable each of the connections within the plurality of interconnections to form the control flow graph with the plurality of node elements, the control flow graph being one of multiple possible control flow graphs implementable by enabling and disabling an arbitrary set of the connections within the plurality of interconnections; and

a plurality of evaluation logic elements coupled to the interconnections and operable to evaluate input data against criteria, the plurality of evaluation logic elements to control one or more transitions between node elements in the control flow graph. (emphasis added)

As set forth above, Claim 1 is a hardware architecture for implementing computational and control flow graphs and includes connectivity control logic to independently enable and disable each connections within multiple interconnections to form a control flow graph with multiple node elements. The control flow graph is one of multiple possible control flow graphs implementable by enabling and disabling an arbitrary set of the connections within the interconnections.

On examination, the Mergard patent may appear to the Examiner to describe state machine evaluation using programmable logic elements with masking of both inputs and state transitions, but Mergard does not disclose how each of the multiple classes of state machines can be realized. For example, Mergard does not describe how non-deterministic finite state automata can be realized, nor how generalized computational and control flow graphs could be implemented.

In contrast, the present invention as claimed builds arbitrary computational and control flow graphs in hardware. The computational control flow graphs needed to implement non-deterministic finite state automata are illustrated in Figure 3(a) of the application. One embodiment of the present invention as claimed includes an apparatus that can be used to implement such graphs. In one embodiment, the apparatus comprises both storage elements (or nodes) and wires (or interconnections) that can be organized into any arbitrary graph through programmable connectivity control. Furthermore, the apparatus provides evaluation logic that is coupled to the control flow graph to enable computational flow through this control flow graph. In this way, non-deterministic finite state automata can be realized. In one embodiment, each node of the graph can be connected to any and every other node in a programmable manner. Also computation flow can transition through each node to any and every other node, and furthermore every node can simultaneously and independently make such a transition in a programmable manner. It is this property that enables the realization of programmable non-deterministic finite state automata.

Therefore, Applicant respectfully submits that Mergard does not disclose connectivity control logic to independently enable and disable each connection within multiple interconnections to form a control flow graph with multiple node elements where the control flow graph is one of multiple possible control flow graphs implementable by enabling and disabling an arbitrary set of the connections within the interconnections. In view of this, Applicant respectfully submits that the present invention as claimed is not anticipated by Mergard.

Additionally, as set forth in Claim 25, the present invention as claimed includes an architecture that enables the realization of larger control flow graphs by hierarchical use of the building blocks described. Mergard does not teach, mention, nor disclose a hierarchical

arrangement as set forth in the claims, or the realization of arbitrary computation and control flow graphs for implementing non-deterministic finite state automata. In view of this, Applicant respectfully submits that the present invention as claimed is not anticipated by Mergard.

Accordingly, Applicant respectfully submits that the rejection under 35 U.S.C. § 102(b) has been overcome by the amendments and the remarks. Applicant submits that claims 1-23 and 25 as amended and Claims 26 and 27 as added are now in condition for allowance and such action is earnestly.

If there are any additional charges, please charge Deposit Account No. 02-2666 for any fee deficiency that may be due.

Respectfully submitted,

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP

Date: 8/30, 2005

By: Michael J. Mallie  
Michael J. Mallie  
Reg. No. 36,591

12400 Wilshire Boulevard  
Seventh Floor  
Los Angeles, California 90025  
(408) 720-8300



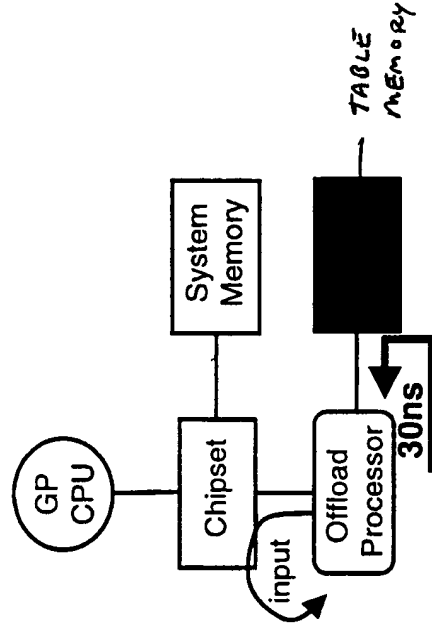
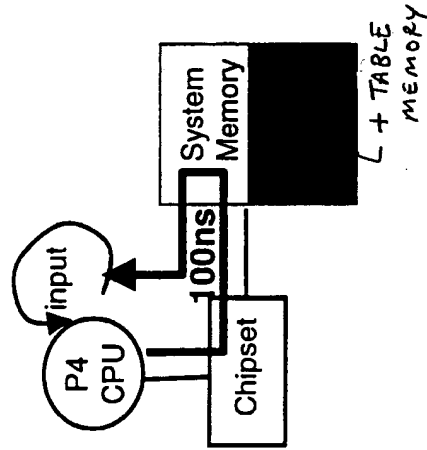
Figure 1(a)

Properties of DFA and NFA techniques used on conventional microprocessors	Storage: Bound on # of States (for an R character Regular Expression)	Evaluation time (for N bytes of Input) [ order of]
Deterministic Finite State Automata or DFA running on a GP CPU	$2^R$ (needs very large memory)	$N$ memory access cycles
Non-Deterministic Finite State Automata or NFA running on a GP CPU	$R$	$R * N$ cpu cache+branch cycles

Figure 1(b)

CPU walking DFA table in DRAM

Coprocessor closer to table in SRAM



Performance on evaluating Regular Expressions on every byte of input stream

1000s of REs @ 100 Mbps

100s of REs @ 280 Mbps

Gigabytes of Memory

100s of MBs of SRAM

## Figure 2

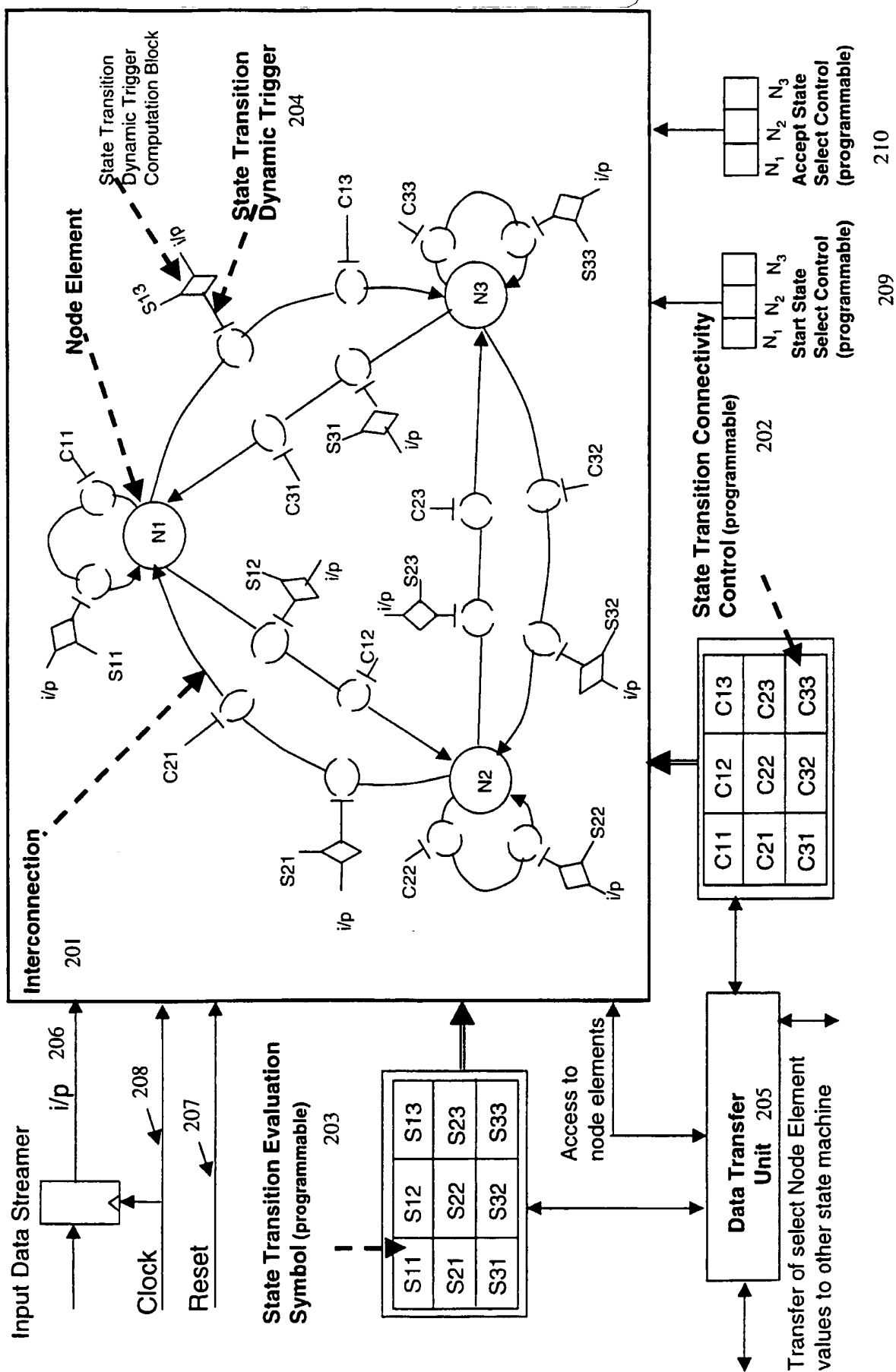




Figure 3(a)

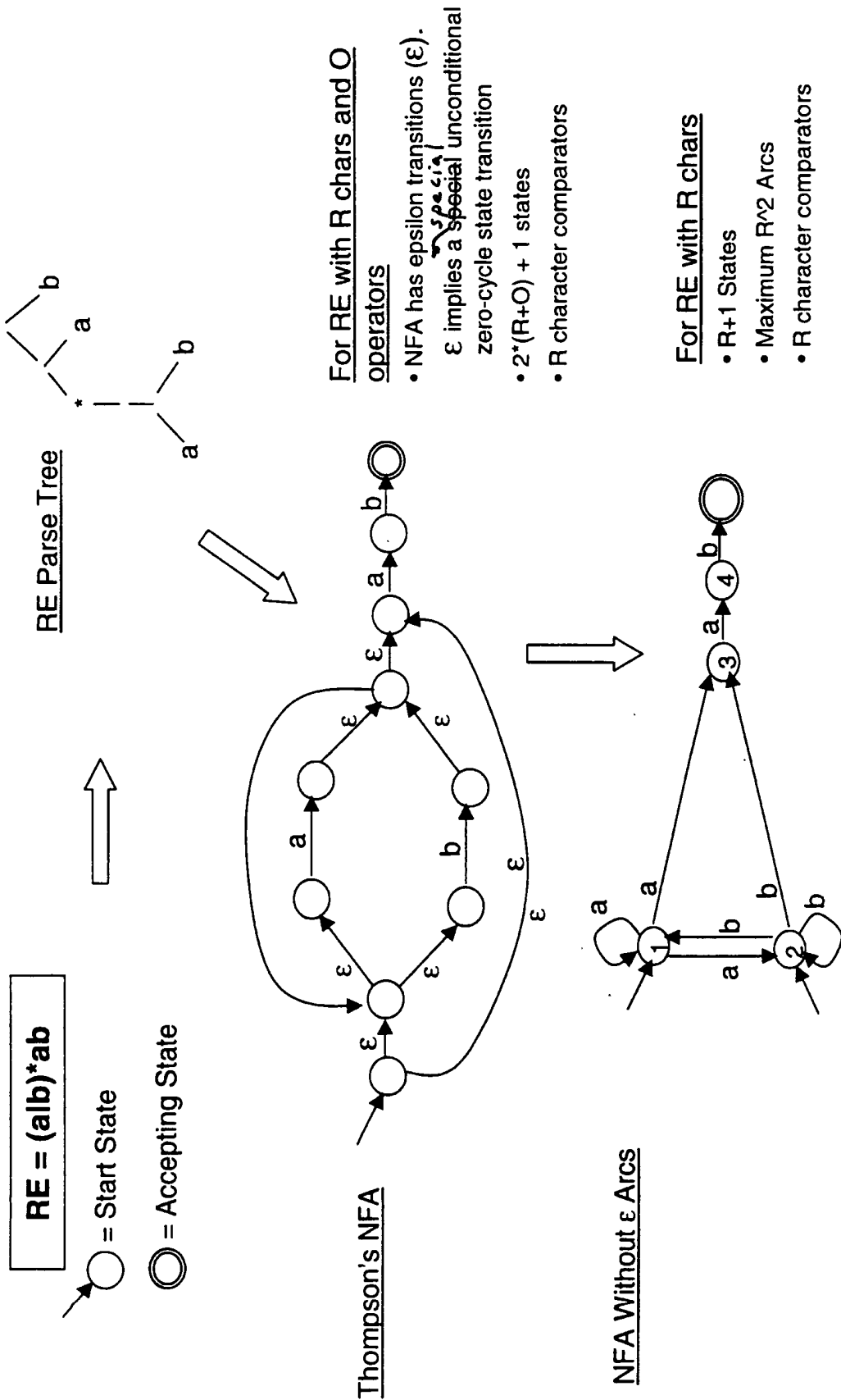


Figure 3(b)

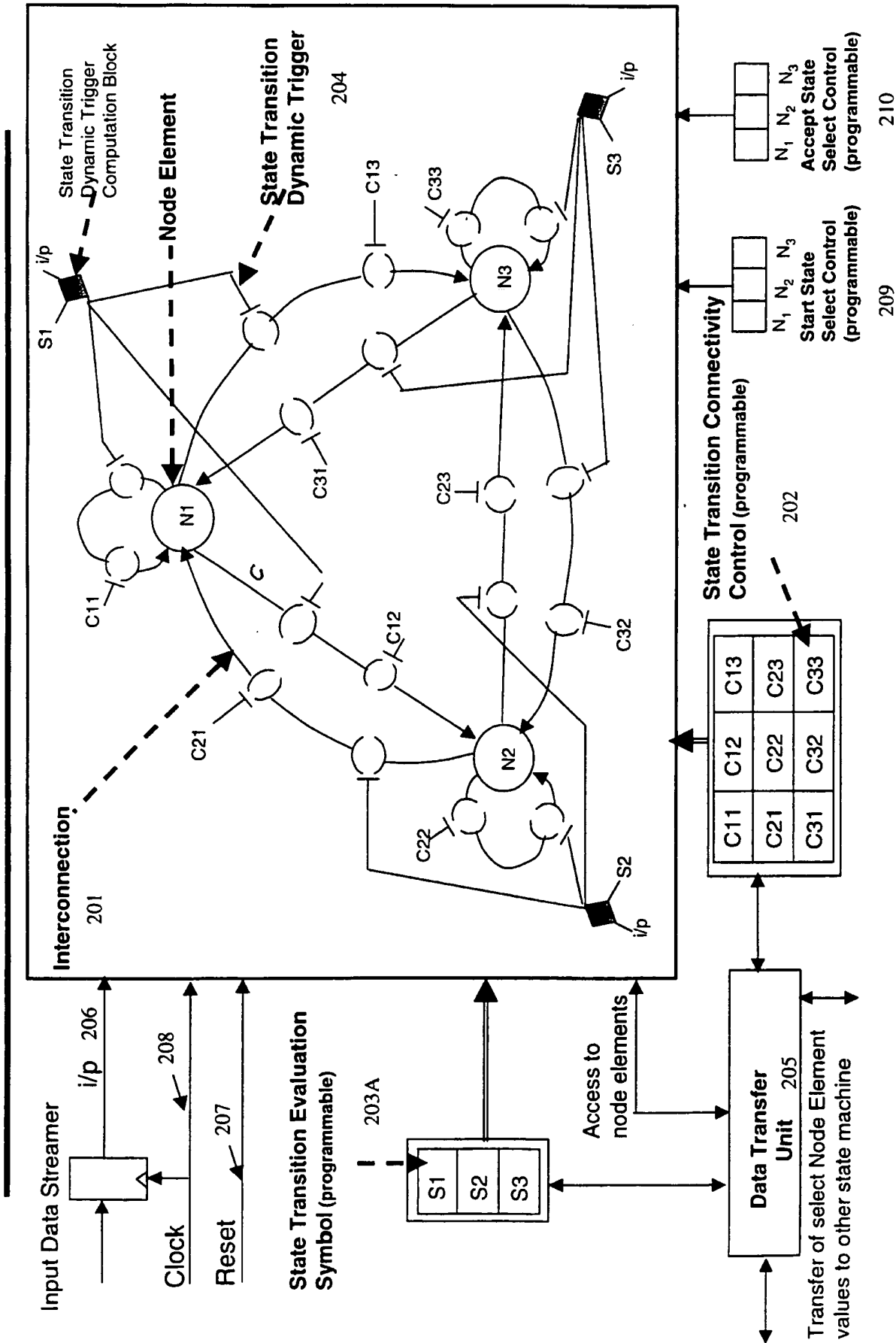


Figure 4

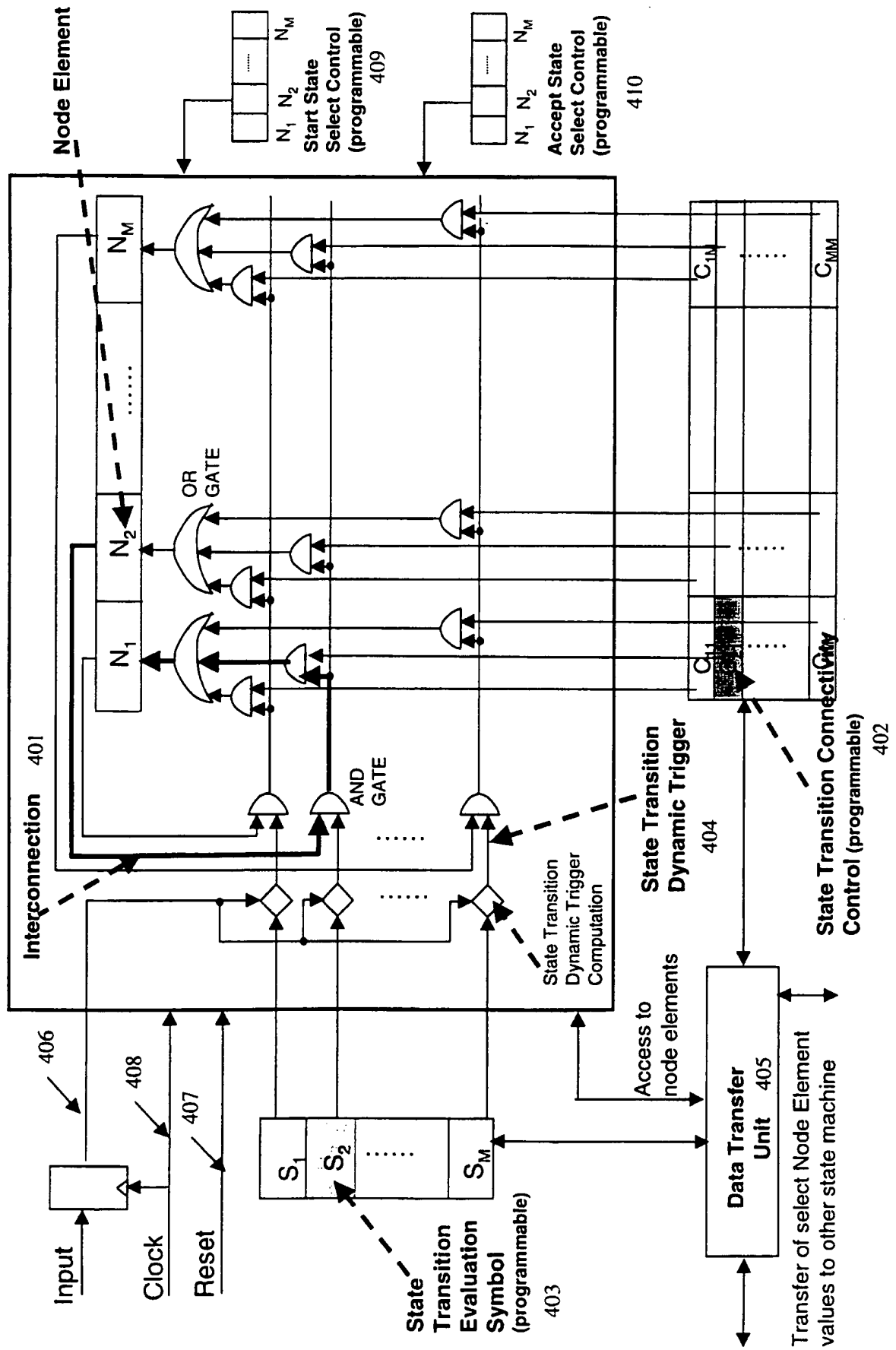


Figure 5

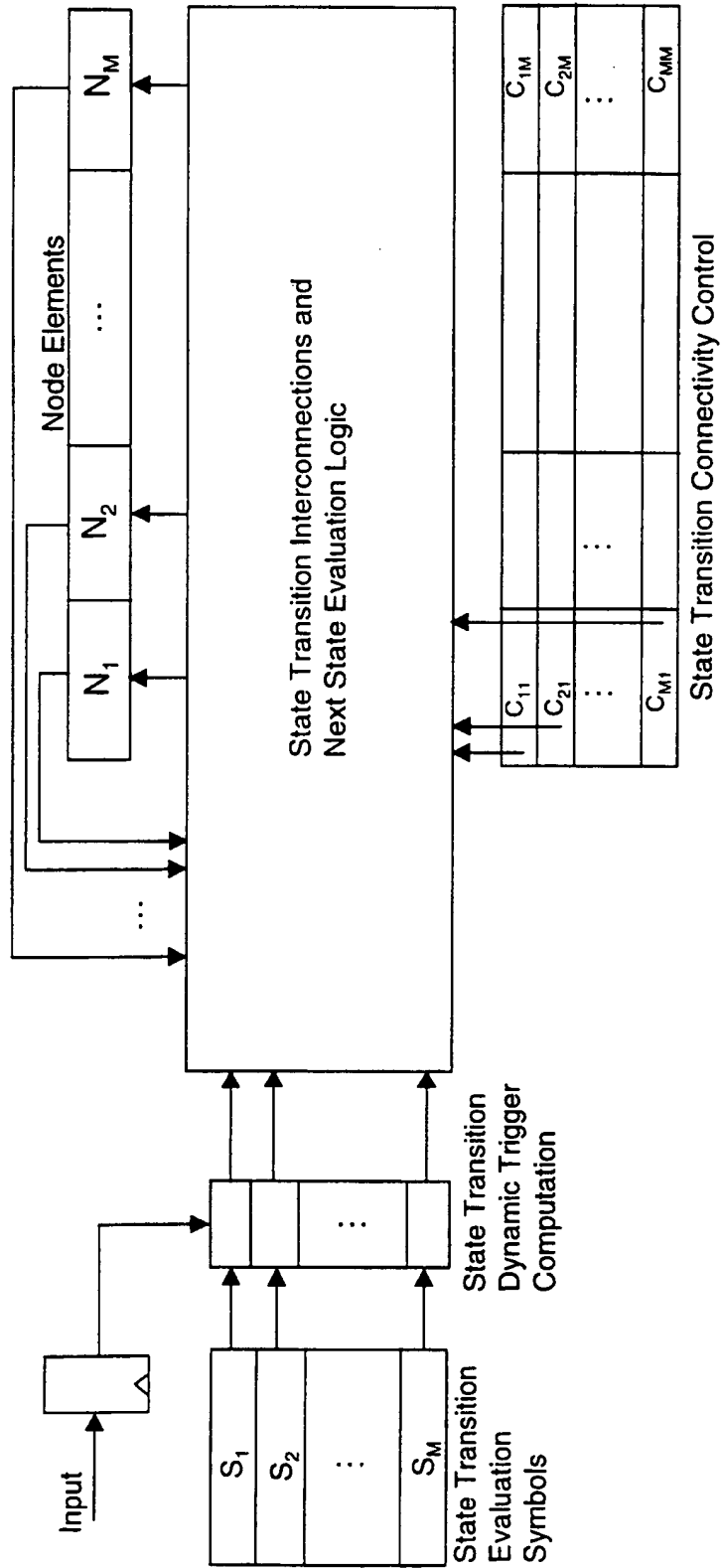


Figure 6  
Programmable Automata Registers (Expression Registers)

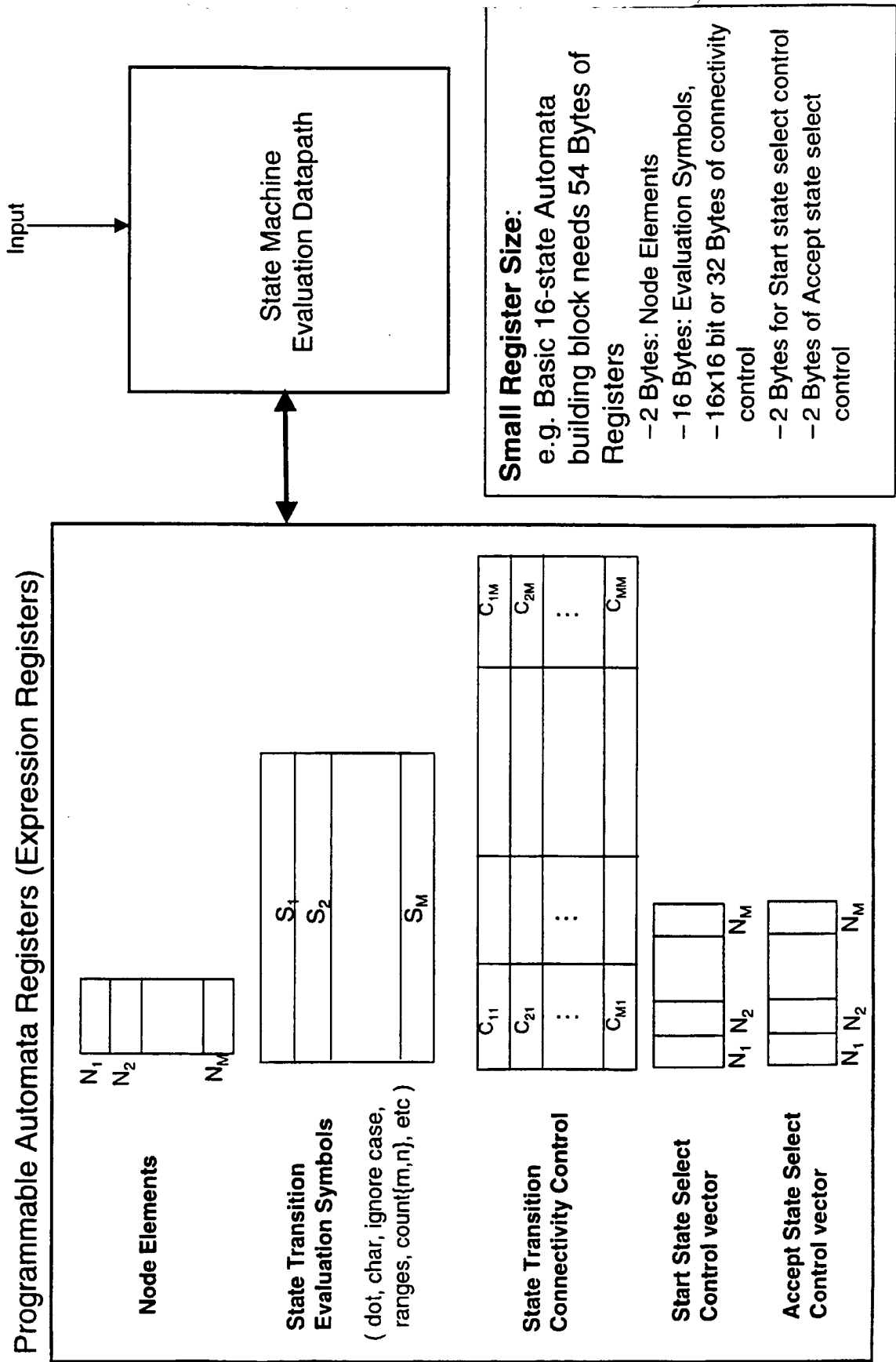
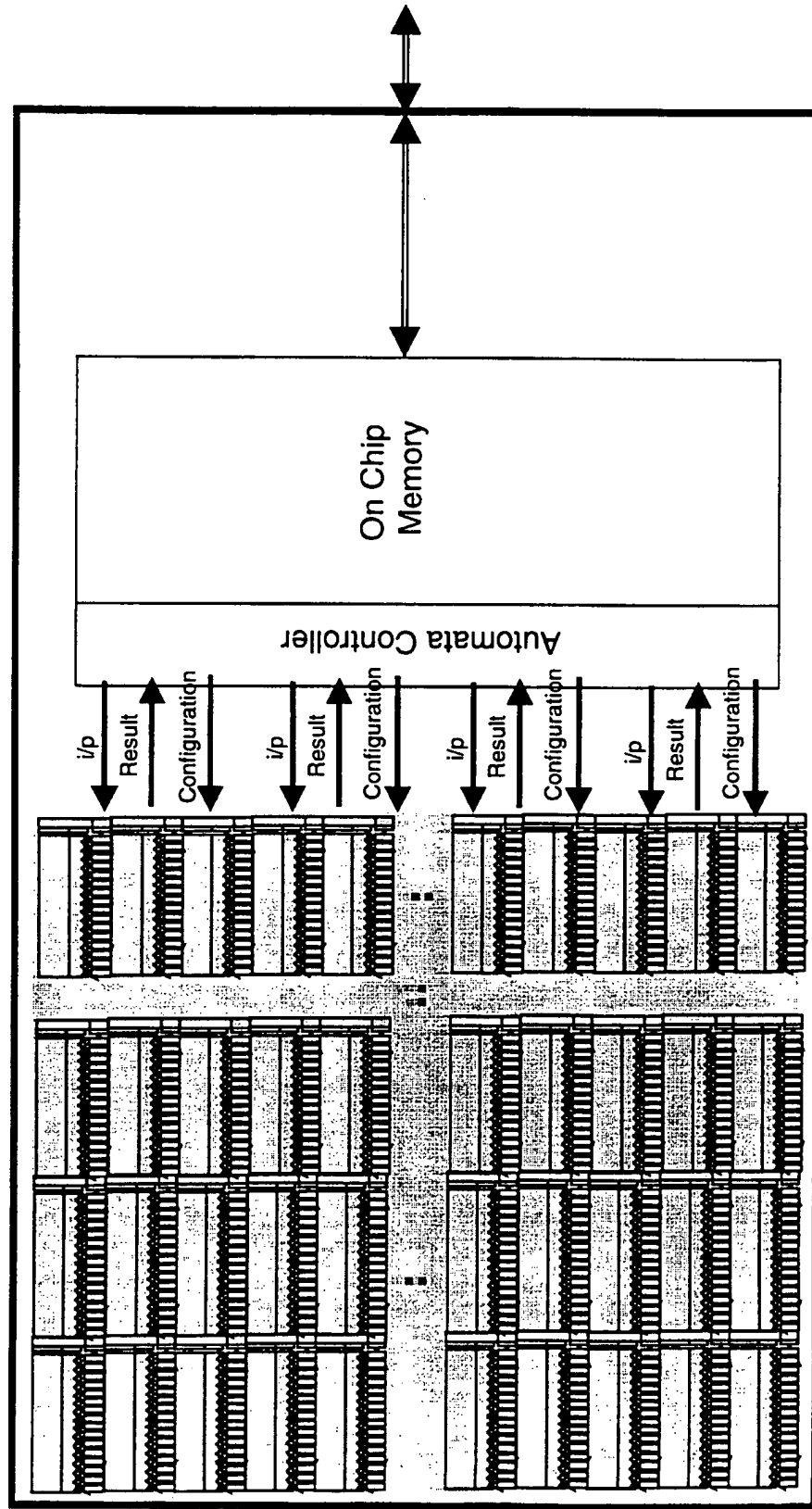


Fig 7

- Simple regular structure enables a high density → dense array of multiple tiles
- Several thousands of automata (organized as multiple rows of tiles) can fit on a single die on 0.13u technology



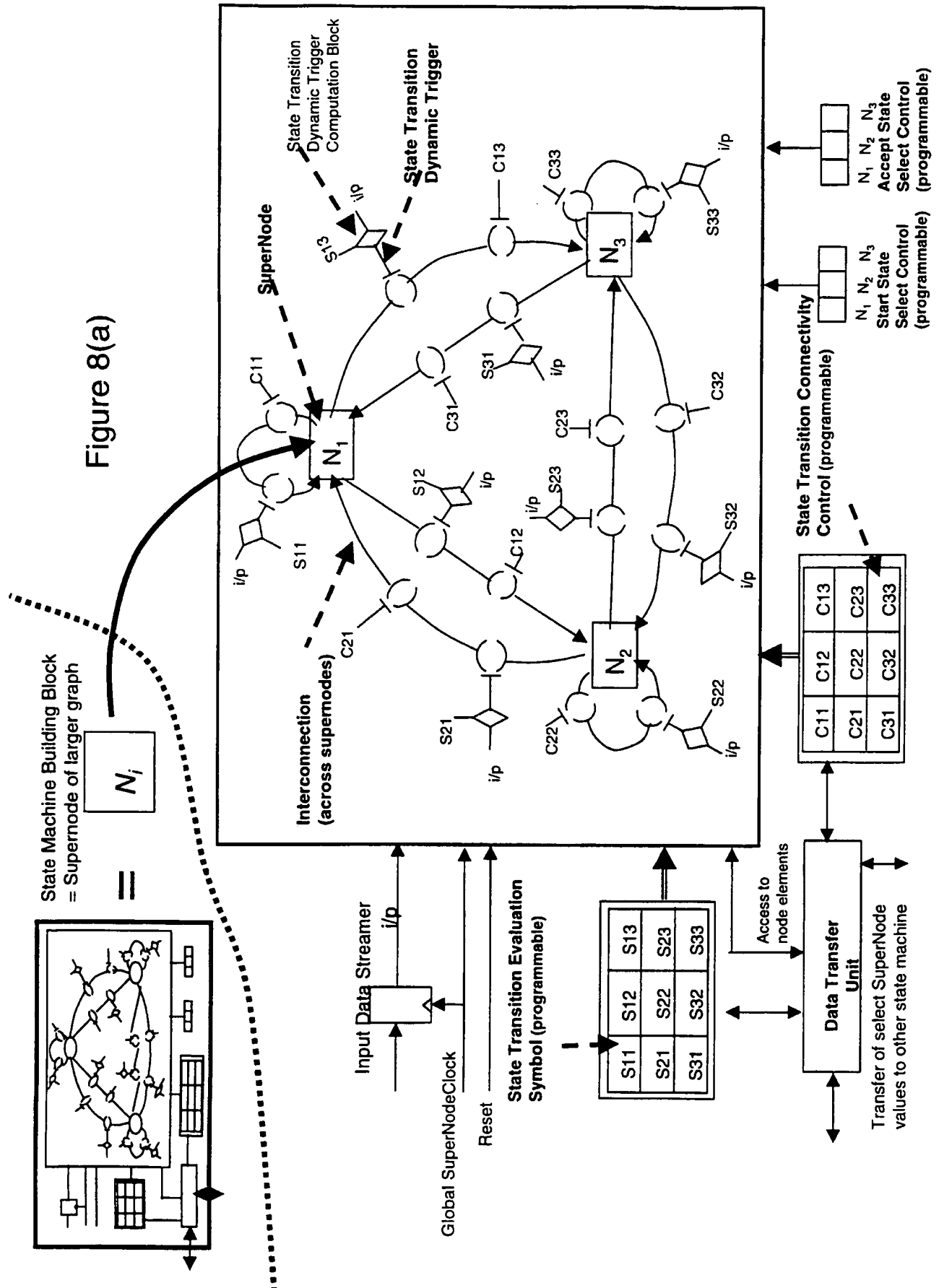
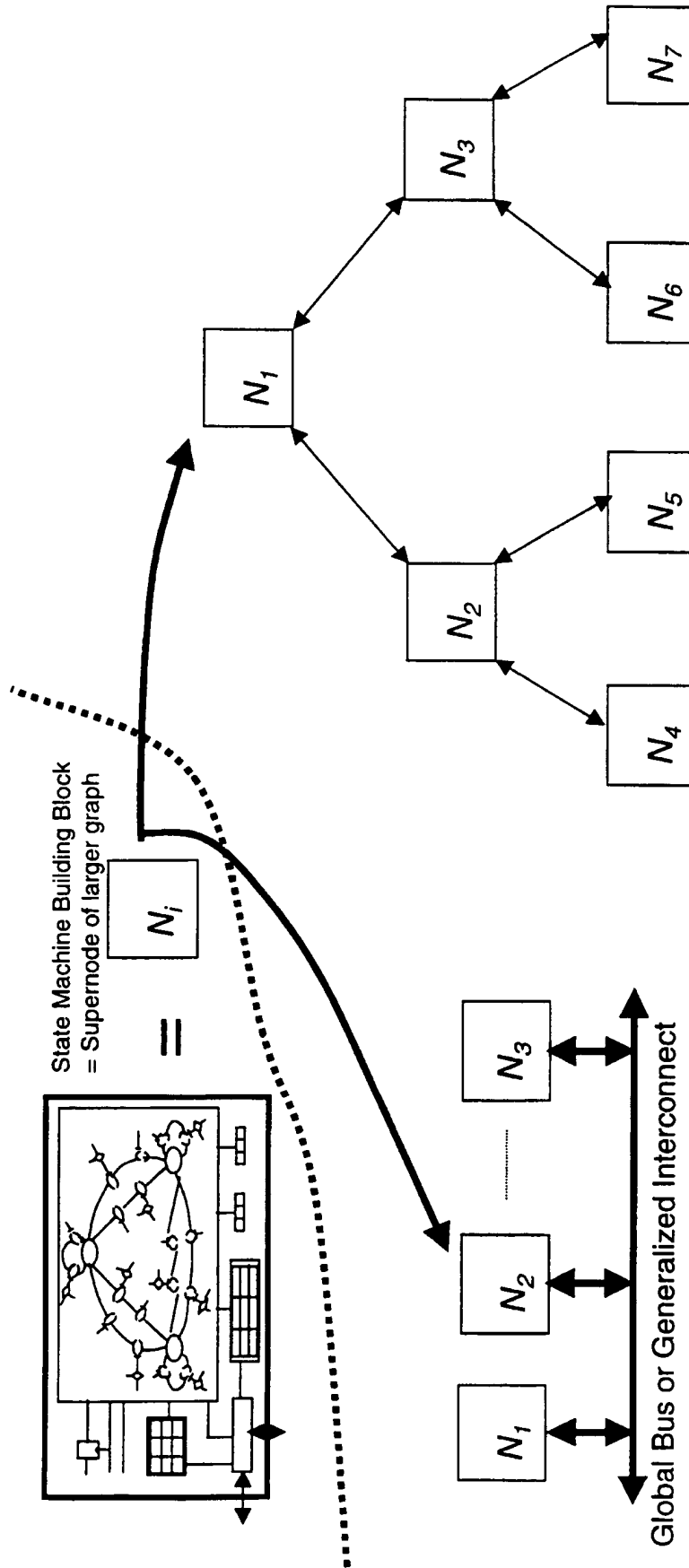


Figure 8(b)



Larger State Machine Realized via Multiple  
State Machine Building Blocks  
Organized as a Tree

Larger State Machine Realized via Multiple  
State Machine Building Blocks  
on a General Interconnect



Figure 9(a)


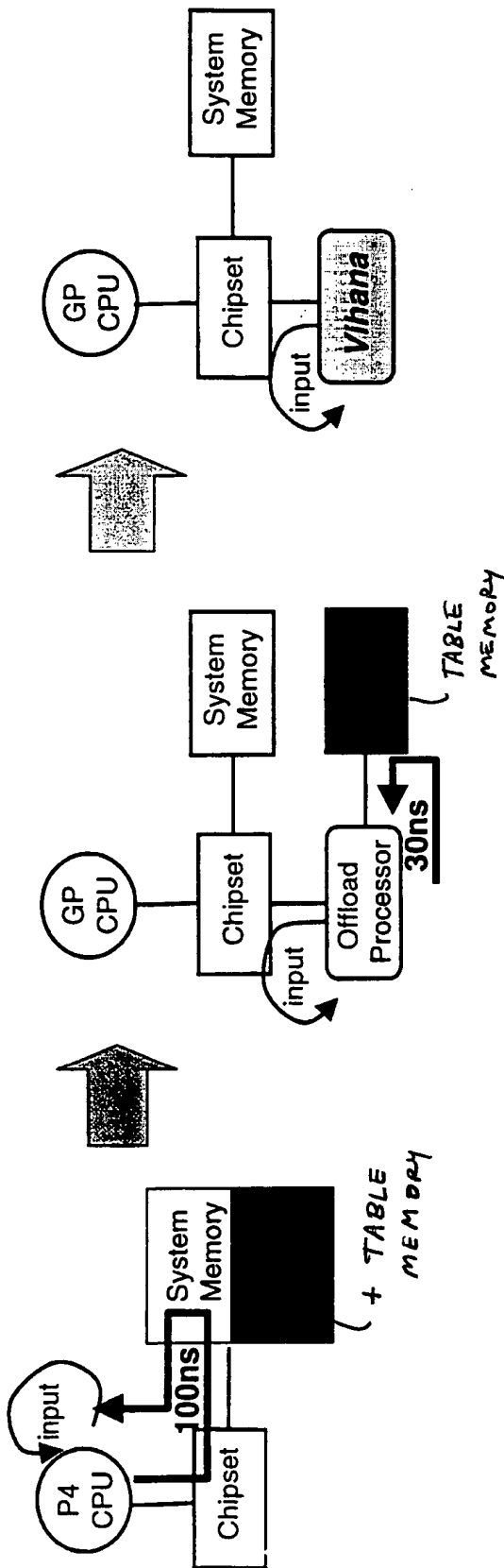
Properties of DFA and NFA techniques used on conventional microprocessors	Storage: Bound on # of States (for R characters)	Evaluation time (for N bytes) [ order of]
Deterministic Finite State Automata or DFA running on a GP CPU	$2^R$ (needs very large memory)	$N$ memory access cycles (~100ns)
Non-Deterministic Finite State Automata or NFA running on a GP CPU	$R$	$R * N$ cpu cache+branch cycles (~4ns)
		
Non-Deterministic Finite State Automata or NFA running on the Apparatus	$R$	$N$ Tight on chip state transition cycle (~1 ns)

Figure 9(b)

Regular Expression Co-processor using Exemplary State machine Architecture

Coprocessor closer to table in SRAM

CPU walking DFA table in DRAM



Perf on REs on every byte

1000s of REs @ 100Mbps

100s of REs @ 280Mbps

1000s of REs @ > 10Gbps

Gigabytes of Memory

100s of MBs of SRAM

No table memory needed

Two orders of magnitude speedup without need for table memory